PAPER • **OPEN ACCESS**

Siamese Networks for Speaker Identification on Resource-Constrained Platforms

Journal of Physics: Conference Series **2622** (2023) 012014 doi:10.1088/1742-6596/2622/1/012014

Siamese Networks for Speaker Identification on Resource-Constrained Platforms

Lim Jun Jie, Muhammad Mun'im Ahmad Zabidi*, Shahidatul Sadiah, Ab Al-Hadi Ab Rahman

Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia

E-mail: munim@utm.my

Abstract. This paper investigates the implementation of a lightweight Siamese neural network for enhancing speaker identification accuracy and inference speed in embedded systems. Integrating speaker identification into embedded systems can improve portability and versatility. Siamese neural networks achieve speaker identification by comparing input voice samples to reference voices in a database, effectively extracting features and classifying speakers accurately. Considering the trade-off between accuracy and complexity, as well as hardware constraints in embedded systems, various neural networks could be applied to speaker identification. This paper compares the incorporation of CNN architectures targeted for embedded systems, MCUNet, SqueezeNet and MobileNetv2, to implement Siamese neural networks on a Raspberry Pi. Our experiments demonstrate that MCUNet achieves 85% accuracy with a 0.23-second inference time. In comparison, the larger MobileNetv2 attains 84.5% accuracy with a 0.32-second inference time. Additionally, contrastive loss was superior to binary cross-entropy loss in the Siamese neural network. The system using contrastive loss had almost 68% lower loss scores, resulting in a more stable performance and more accurate predictions. In conclusion, this paper establishes that an appropriate lightweight Siamese neural network, combined with contrastive loss, can significantly improve speaker identification accuracy, and enable efficient deployment on resource-constrained platforms.

1. Introduction

Speaker identification is often used in biometric security systems to determine person's identity. It works by comparing the unknown speaker's audio to the models of all enrolled speakers. The best-matching speaker is the one who is most likely to be the person speaking. Speaker identification is different from speaker verification, which only checks if the speaker's identity matches the claimed identity. Speaker identification requires N comparisons to identify a speaker from a group of N people, while speaker verification only requires one comparison. The following are the key steps in speaker identification: (1) extracting features from the audio, (2) comparing those features to a speaker model in the database of known speakers, and (3) making a decision based on the comparison.

A successful speaker identification largely depends on the model used on the speaker's voice. This model can be created using various techniques such as Gaussian mixture models (GMM), Hidden Markov models (HMM), support vector machines (SVM), or deep neural networks (DNN). Lately, DNN have been popular due to their good performance.

In a typical DNN structure, the model learns to classify data using a provided dataset and generates a prediction probability. However, to achieve high prediction probabilities, the CNN

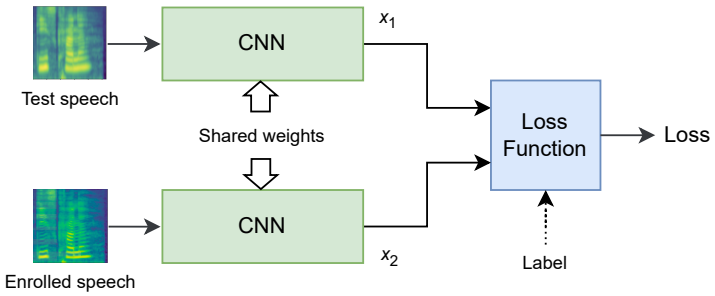

Figure 1. Architecture of Siamese Neural Network

must be trained on a sufficiently large dataset. This poses a challenge when a new training dataset is required to identify a new speaker, as creating and utilizing it can be time-consuming.

Siamese neural networks (SNN), shown in Figure 1, are particularly well-suited for speaker identification tasks because they specialize in learning the similarity or dissimilarity between two input samples [1]. For speaker identification, an SNN compares the features of an unknown speech signal against reference features from a speech database, determining the best match by comparing similarity scores. Compared to other methods, Siamese networks require fewer training samples while maintaining high accuracy. They are also more robust to variations in speech signals and can handle large speech datasets.

The speaker identification process involves comparing an unknown speaker's utterance to a database of enrolled speakers. If the utterance matches the database above a certain threshold, the claim is accepted, otherwise it is rejected. The accuracy of speaker identification depends on choosing an appropriate threshold value. A low threshold can lead to inaccurate identification, while a high threshold can make identification difficult [2].

Feature extraction is essential for preprocessing speaker identification data. It reduces dimensionality by dividing the raw data into smaller, more manageable groups. This aids training by extracting critical information from speech waves while reducing model complexity [3, 4]. The extracted features are then input to neural networks for model training.

This paper describes an embedded implementation of speaker identification using Siamese Neural Networks. It was deployed on the Raspberry Pi 4 Model B. Due to the Pi's resource constraints, identifying suitable SNN subnetworks is crucial to ensure acceptable real-time execution. In addition to well-known lightweight networks like MobileNetv2 and SqueezeNet, MCUNet, which runs on NAS, is also used as one of SNN's subnetworks. A wake word detector is used in conjunction with the speaker identification model to meet the requirements of the real-time use case. A GUI is created using Qt, an open-source, multi-platform framework, to allow users to interact with the speaker identification system in real-time [5].

2. Research Method

2.1. Dataset

The dataset used for training is the VoxCeleb2 open-source media dataset that consists of *.m4a files. It has more than a million utterances from more than 6,000 speakers [6]. It is chosen due to its variation in noise that can mimic a noisy environment [7]. Thus, the speaker identification model that is trained by using this dataset will have good noise robustness real-time scenarios. The raw data is divided into training and testing parts in the ratio 8:2 respectively. This means that 80% of the dataset is used to train the model, while 20% of the dataset is used to test the model's prediction accuracy. This process can also keep the learning model from overfitting its training data. The feature extraction process then runs on the partitioned dataset.

2.2. Feature Extraction

Feature extraction is a dimensionality reduction technique that can be used to reduce a large set of raw data into smaller groups for processing. New features will then be generated from the existing features that can be found in the original dataset [8]. It is one of the essential phases of constructing the speaker identification model for extracting meaningful data from speech waves and reducing model complexity. This extracted information will be loaded into the appropriate neural network architectures for model training.

Several feature extraction techniques can be used in the speaker identification system, each with its own pros and cons. For instance, PLP (perceptual linear prediction) and MFCC (Mel-frequency cepstral coefficients) have been shown to yield superior results due to their architecture's close resemblance to human perception of voice. While LPC (linear predictive coding) is not based on the human auditory system, it is well-suited for systems that require audio communication over long distances [9].

GFCC (Gammatone frequency cepstral coefficients) is a newer technique that shares many similarities with MFCC. While MFCC uses triangular Mel-frequency filters and compresses the signal's dynamic range through logarithmic nonlinearity, GFCC emphasizes spectral valleys with a cubic root nonlinearity and employs a gammatone filter bank. GFCC is particularly beneficial for low-frequency sensitive applications like music and animal sounds. However, it can be more vulnerable to noise in specific scenarios [10, 11]. In general, MFCC is more commonly used in speech-related applications due to its accessibility and good performance with traditional machine learning methods.

For speech related applications using deep learning, Mel-spectrograms are better than MFCC and GFCC. Unlike MFCC, the Mel-spectrogram simplifies the process by eliminating the need for DCT computation, as shown in Figure 2. This simplification allows for the retention of the complex speech signal representation necessary for CNNs, while simultaneously reducing the overall computational burden. Although it is possible to combine multiple feature extraction techniques to leverage their respective strengths, Mel-spectrograms alone are often enough to train accurate models [11].

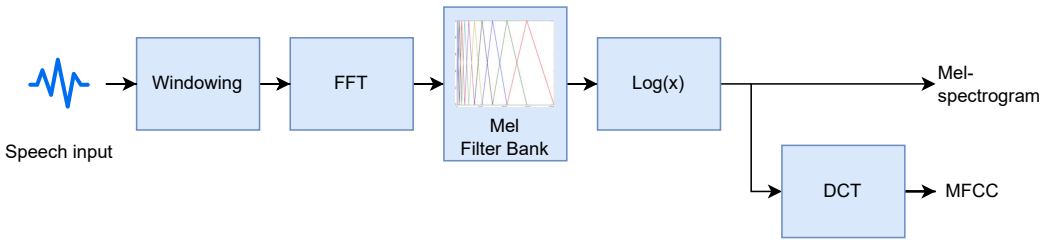

Figure 2. Derivation of MFCC and Mel-Spectrogram

2.3. Loss Function

A loss function evaluates the effectiveness of a neural network in modeling the training data by comparing the difference between target and predicted output values. Based on the neural network model's loss function performance, the hyperparameters of the neural network are adjusted to achieve the lowest possible average loss score. This ensures that the neural network model can adapt to the task at hand. SNN can utilize several loss functions, including binary cross-entropy, contrastive loss, triplet loss, and constellation loss.

Binary cross-entropy is a loss function that can adapt to speaker identification purposes as it compares each predicted probability to the actual class output of false (0) or true (1), indicating whether the target is the correct speaker or not. The score is then calculated by penalizing the

respective probabilities based on how far they are from the predicted value, measuring how close the predicted value is to the actual value.

The contrastive loss has been shown to outperform the binary cross-entropy loss function in SNN for speaker identification tasks, as it excels at distinguishing between two objects [12]. Since the primary purpose of SNN is to compare the similarity of different instances rather than classifying a single instance, the contrastive loss is more appropriate for this scenario. It operates by bringing similar samples closer together and pushing distinct samples farther apart when presented with two input samples that are either similar or different. The calculation of the contrastive loss follows the formula below. Y can be set to 0 if the samples are similar and 1 otherwise. Based on the expression of contrastive loss, the term $||x_i - x_j||^2$ that corresponds to the Euclidean distance will be minimized in case of the samples are coming from similar groups else the term $max(0, ||x_i - x_j||^2)$ will be minimized in case of the samples are coming from dissimilar groups [13]. This is processed by the equation through the search for the correct pair of the inputs, x_i and x_j, so the term of $||x_i - x_j||^2$ that corresponds to their respective Euclidean distances can be achieved. In short, the loss score is aimed to be lowered by minimizing the term of $||x_i - x_j||^2$ for the result of the similar group while minimizing the term of $max(0, ||x_i - x_j||^2)$ for the result of dissimilar group.

$$L = (1 - Y) \times ||x_i - x_j||^2 + Y \times max(0, m - ||x_i - x_j||^2) \quad (1)$$

where

L	=	Loss score
Y	=	True output
x_i	=	Observation vector from input
x_j	=	Target vector from training dataset
$\|\|x_i - x_j\|\|^2$	=	Predicted output
m	=	Hyperparameter that specifies the lower bound distance between dissimilar samples.

2.4. Model Training

Siamese neural networks offer several advantages for speaker identification, notably high accuracy, invariance to environmental changes, and invariance to changes in the speaker's voice, such as age, accent, and emotion [14]. Once trained, the speaker identification model based on SNN does not need to be retrained for identifying a new speaker. SNN's pattern matching analysis is performed against the reference features of the speech database, and the best match is identified by comparing the similarity scores. The training process of SNN is carried out by learning from semantic similarity to do the comparison rather than learning the features directly from the large-scale dataset to do the classification as CNN does. Therefore, SNN is robust to the dataset as it focuses on learning from the semantic similarity and only a few samples per class are required for it to learn the embeddings which place the same classes together.

Several high performance neural network architectures, such as VGG16 and ResNet50, have been used as subnetworks to speaker identification models [2, 15, 16]. However, these networks have high resource consumption and are unsuitable for deployment on embedded systems. Table 1 lists several lightweight CNNs compared to VGG and ResNet.

MobileNetv2 serves as the baseline subnetwork in our work. After feature extraction, the SNN architecture is first implemented using a customized MobileNetv2 subnetwork with no classification layer. Input spectrograms are fetched into the subnetworks, and a custom distance function is used to connect both subnetworks and measure the difference between outputs. This is followed by a flatten layer and a dense layer utilizing the sigmoid activation function, outputting values between 0 and 1. The model is trained using binary cross-entropy as the loss function and Adam as the optimizer with a learning rate of 3e-4. A batch size of 32 is used, and

Table 1. Selected CNN architectures for speaker identification.

Network	Number of Parameters (M)	Memory Used (MB)	Top-5 ImageNet accuracy
VGG16 [15]	138.3	528	92.9%
ResNet50 [16]	25.6	98	93.6%
MobileNetV2 [17]	3.4	14	71.9%
ShuffleNetV2 [18]	1.4	5	69.4%
SqueezeNet [19]	1.2	5	59.5%
MCUNetV2 [20]	1.1	4	74.3%

the model is trained for 40 epochs with 1000 steps per epoch. The training process is performed using NVIDIA GeForce GTX1080 Ti with 11GB GPU memory.

In the next phase of the work, contrastive loss was used as the loss function. Contrastive loss was found to be more suitable for SNN as it pulls clusters of points belonging to the same class closer in the embedding space while pushing away clusters of points belonging to different classes [21]. Thereafter, the model is retrained using contrastive loss for more effective information labeling.

SqueezeNet is then used as a subnetwork of SNN due to its lower parameter count [19]. The batch size is increased to 64 to take advantage of available GPU memory, while all other parameters remain unchanged. The model is trained for 40 epochs with 1000 steps per epoch using contrastive loss.

To further enhance the architecture, the subnetwork of the SNN is replaced with the neural network structure of MCUNet, designed to fit within 256 kB memory constraints. MCUNet optimizes accuracy, memory usage, and energy efficiency by combining the efficient neural architecture of TinyNAS with the lightweight inference engine of TinyEngine [20]. The two-stage neural architecture search of TinyNAS was used without inferring the model using TinyEngine. TinyNAS creates a specialized network architecture after optimizing it to fit the resource constraints. To train the speaker identification model using TinyNAS of MCUNet, the batch size remains at 64 while other parameters remain unchanged, and the model is trained for 40 epochs with 1000 steps per epoch using contrastive loss.

3. Results and Discussion

3.1. Training Results

In addition to accuracy, a model's performance can be evaluated with a loss score, which measures the difference between predicted and actual values. On the MobileNetV2 subnetwork, both binary cross-entry and contrastive loss functions were used, with contrastive loss retained for the remaining experiments. Due to GPU limits, the SNN batch size was set to 32 and training converged around the 40th epoch. The same was true for the other subnetworks, so training was stopped at the 40th epoch to prevent overfitting.

Figures 4 to 7 show the training results for a SNN with MobileNetv2 as the subnetwork. The plots also compare the training results using binary cross-entropy loss and contrastive loss as the loss functions, respectively. Figures 8 and 9 show the training results using SqueezeNet as the subnetwork, while Figures 10 and 11 show the training results using MCUNet. The training results are summarized in Table 2.

In conclusion, the speaker identification model using MCUNet256kb as the SNN subnetwork achieved the highest accuracy on the Raspberry Pi 4. The SqueezeNet model demonstrated the fastest inference time. Regardless of the SNN subnetwork type, all speaker identification models can be deployed on the Raspberry Pi 4. MCUNet256kb has fewer parameters than

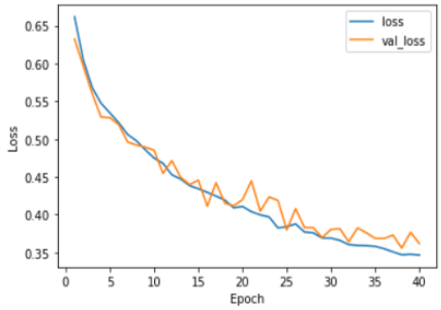

Figure 3. MobileNetV2 Loss vs Epoch using binary cross-entropy loss function.

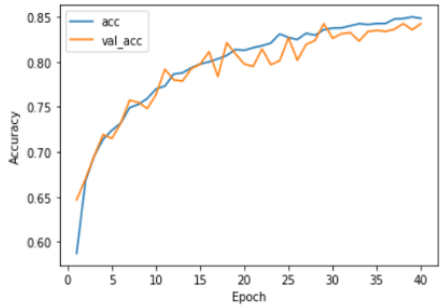

Figure 4. MobileNetV2 Accuracy vs Epoch using binary cross-entropy loss function.

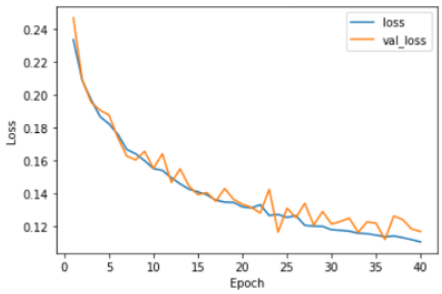

Figure 5. MobileNetV2 Loss vs Epoch using constrastive loss function.

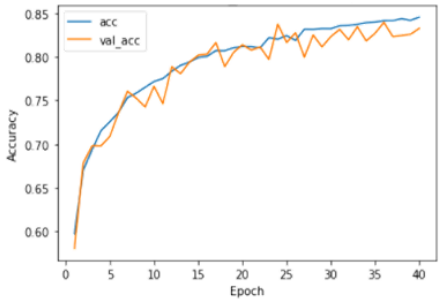

Figure 6. MobileNetV2 Accuracy vs Epoch using constrastive loss function.

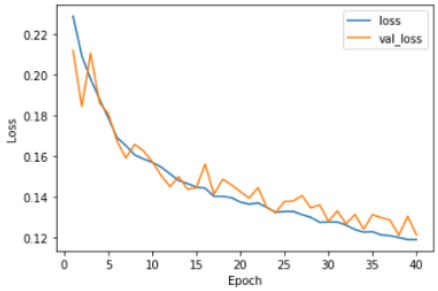

Figure 7. SqueezeNet Loss vs Epoch using constrastive loss function.

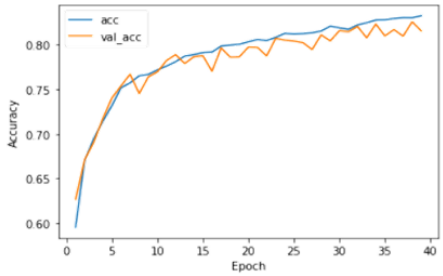

Figure 8. SqueezeNet Accuracy vs Epoch using constrastive loss function.

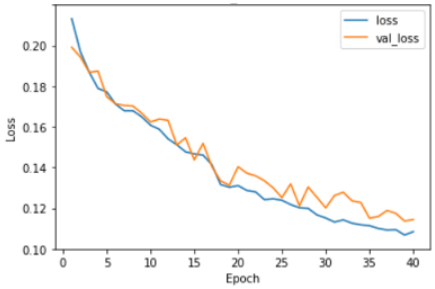

Figure 9. MCUNet Loss vs Epoch using constrastive loss function.

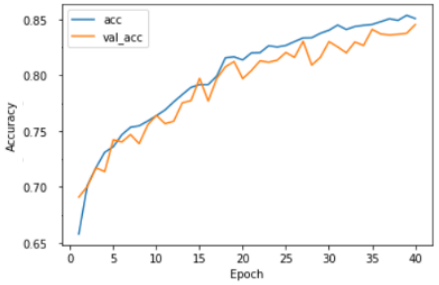

Figure 10. MCUNet Accuracy vs Epoch using constrastive loss function.

Table 2. Summary of training results.

Network	Loss Function	Batch Size	Accuracy (%)	Loss	Inference Time (s)
MobileNetv2	Binary cross entropy	32	84.50	0.3466	0.32
MobileNetv2	Contrastive loss	32	84.80	0.1103	0.32
SqueezeNet	Contrastive loss	64	83.30	0.1188	0.14
MCUNet256kb	Contrastive loss	64	85.00	0.1084	0.23

SqueezeNet but takes longer for inference due to its deeper network and more operations. By using MobileNetv2 as a baseline, it is possible to build a speaker identification model with either higher accuracy or faster inference time by making a trade-off between these two factors. However, the differences in each factor are not significant. This work highlights that MCUNet, a neural architecture search-based model, can be successfully executed on microprocessors with high accuracy and low resource consumption.

3.2. Embedded Deployment

The Raspberry Pi 4 Model B was selected as the platform for deploying the speaker identification model, considering its available resources [22, 23]. The Raspbian 64-bit operating system was installed first, followed by the libraries needed to run Python scripts. The models were trained on the GPU, and the model parameters were saved in HDF5 file format before being loaded into the Raspberry Pi for deployment. A hands-free GUI was created with the PyQt5 library to provide test inputs to the speaker identification model on the Raspberry Pi 4 [5].

To enhance accuracy and reduce power consumption, a two-stage speaker identification system is created by integrating a simple wake word detector with the speaker identification model. The first stage is the always-on wake word detector, while the second stage performs the actual speaker identification process, triggered only after the correct wake word is detected [24]. The wake word detector is built using the Picovoice end-to-end edge AI platform, with "picovoice" serving as the wake word [25]. Once the wake word is detected, the speaker identification stage is activated, and the result is displayed after inference is completed.

4. Conclusion

In conclusion, this paper presents a study on the implementation of a lightweight Siamese neural network (SNN) for speaker identification on resource-constrained platforms. The results demonstrate that SNNs, specifically using the MCUNet subnetwork, can achieve high accuracy and fast inference times on embedded systems like the Raspberry Pi. Furthermore, the contrastive loss function is found to outperform binary cross-entropy loss in the SNN for speaker identification tasks. This research establishes that an appropriate lightweight SNN, combined with contrastive loss, can significantly improve speaker identification accuracy and enable efficient deployment on resource-constrained platforms. Future work could explore the performance of SNNs with other lightweight subnetworks and investigate additional loss functions for further improvements in speaker identification systems on embedded platforms.

Acknowledgement

The authors would like to thank UTM for the funding of this project, with the Flagship CoE/RG research grant number Q.J130000.5023.10G05, Q.J130000.5023.10G06. and Q.J130000.5023.10G09.

References

[1] Mohammad Shorfuzzaman and M Shamim Hossain. MetaCOVID: A siamese neural network framework with contrastive loss for n-shot diagnosis of COVID-19 patients. *Pattern recognition*, 113:107700, 2021.
[2] Bubaker Bushofa and Naji Bazina. User authentication based on his/her speech. In *3rd Intl Conf on Machine Learning and Computer Science (IMLCS'2014)*, 01 2014.
[3] Ivette Vélez, Caleb Rascon, and Gibrán Fuentes-Pineda. One-shot speaker identification for a service robot using a CNN-based generic verifier. *arXiv:1809.04115*, 2018.
[4] Arzo Mahmood and Utku Köse. Speech recognition based on convolutional neural networks and mfcc algorithm. *Advances in Artificial Research*, 1:6–12, 01 2021.
[5] Xibao Wang, Ge Li, and Peng Wang. Qt-based cross-platform design of management system for distributed real-time simulation platform. In *2015 5th International Conference on Computer Sciences and Automation Engineering (ICCSAE 2015)*, pages 856–861. Atlantis Press, 2016.
[6] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. VoxCeleb2: Deep speaker recognition. In *Interspeech 2018*. ISCA, sep 2018.
[7] Umair Khan and Javier Hernando. The UPC speaker verification system submitted to VoxCeleb speaker recognition challenge 2020 (VoxSRC-20). *arXiv:2010.10937*, 10 2020.
[8] Wamidh K. Mutlag, Shaker K. Ali, Zahoor M. Aydam, and Bahaa H. Taher. Feature extraction methods: A review. In *Journal of Physics Conference Series*, volume 1591 of *Journal of Physics Conference Series*, page 012028, July 2020.
[9] Jaison Joshy and Koj Sambyo. A comparison and contrast of the various feature extraction techniques in speaker recognition. *Int J Signal Process Image Process and Pattern Recognit*, 9(11):99–108, 2016.
[10] Xiaojia Zhao and Deliang Wang. Analyzing noise robustness of MFCC and GFCC features in speaker identification. *Proc IEEE Int Conf Acoust Speech Signal Process*, pages 7204–7208, 2013.
[11] Medikonda Jeevan, Atul Dhingra, Madasu Hanmandlu, and Bijaya Ketan Panigrahi. Robust speaker verification using GFCC based i-vectors. In *Proc Intl Conf on Signal, Networks, Computing, and Systems (CSNCS 2016)*, 2017.
[12] C Vimala and V Radha. Suitable feature extraction and speech recognition technique for isolated Tamil spoken words. *Int J Comput Sci Inf Technol*, 5(1):378–383, 2014.
[13] Alfonso Medela and Artzai Picon. Constellation loss: Improving the efficiency of deep metric learning loss functions for the optimal embedding of histopathological images. *Journal of Pathology Informatics*, 11(1):38, 2020.
[14] Amirhossein Hajavi and Ali Etemad. Siamese capsule network for end-to-end speaker recognition in the wild. In *Proc IEEE Int Conf Acoust Speech Signal Process*, pages 7203–7207. IEEE, 2021.
[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc IEEE Conf Computer Vision and Pattern Recognition*, pages 770–778, 2016.
[17] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit*, pages 4510–4520, 2018.
[18] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proc European Conf on Computer Vision (ECCV)*, pages 116–131, 2018.
[19] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1MB model size. *arXiv:1602.07360*, 2016.
[20] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, 33:11711–11722, 2020.
[21] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc., 2020.
[22] Gerardo José Ginovart-Panisello, Ester Vidaña-Vila, Selene Caro-Via, Carme Martínez-Suquía, Marc Freixes, and Rosa Ma Alsina-Pagès. Low-cost WASN for real-time soundmap generation. *The 8th Int Symposium on Sensor Science*, May 2021.
[23] Mohammad Shahrul Izham Sharifuddin, Sharifalillah Nordin, and Azliza Mohd Ali. Comparison of CNNs and SVM for voice control wheelchair. *IAES Int Journal Artificial Intelligence*, 9(3):387, 2020.
[24] Christin Jose, Yuriy Mishchenko, Thibaud Senechal, Anish Shah, Alex Escott, and Shiv Vitaladevuni. Accurate detection of wake word start and end using a CNN. *arXiv:2008.03790*, 2020.
[25] Alireza Kenarsari, Ian Lavery, and Reza Rostam. Picovoice Porcupine, 2022. https://github.com/Picovoice/porcupine.